Applications of Face Detection:

1. Cameras that make sure faces are focused before you take a picture
2. Facebook tags people automatically once you upload a picture
3. Some shows like CSI used to identify “bad guys” from security footage
4. Unlocking your phone -
   - You look at your phone, and it extracts your face from an image (face detection).
   - Then, it compares the current face with the one it saved before during training and checks if they both match.

Theory of **Face Detection Classifiers:**

1. A computer program that decides whether an image is positive image (face image) or negative image (non-face image) is called a **classifier**.
2. A classifier is trained on hundreds of thousands of face and non-face images to learn how to classify a new image correctly.

OpenCV -> two classifiers :

1. Haar Classifier
2. LBP Classifier

Both of these classifiers process images in gray scales, basically because we don’t need color information to decide if a picture has a face or not.

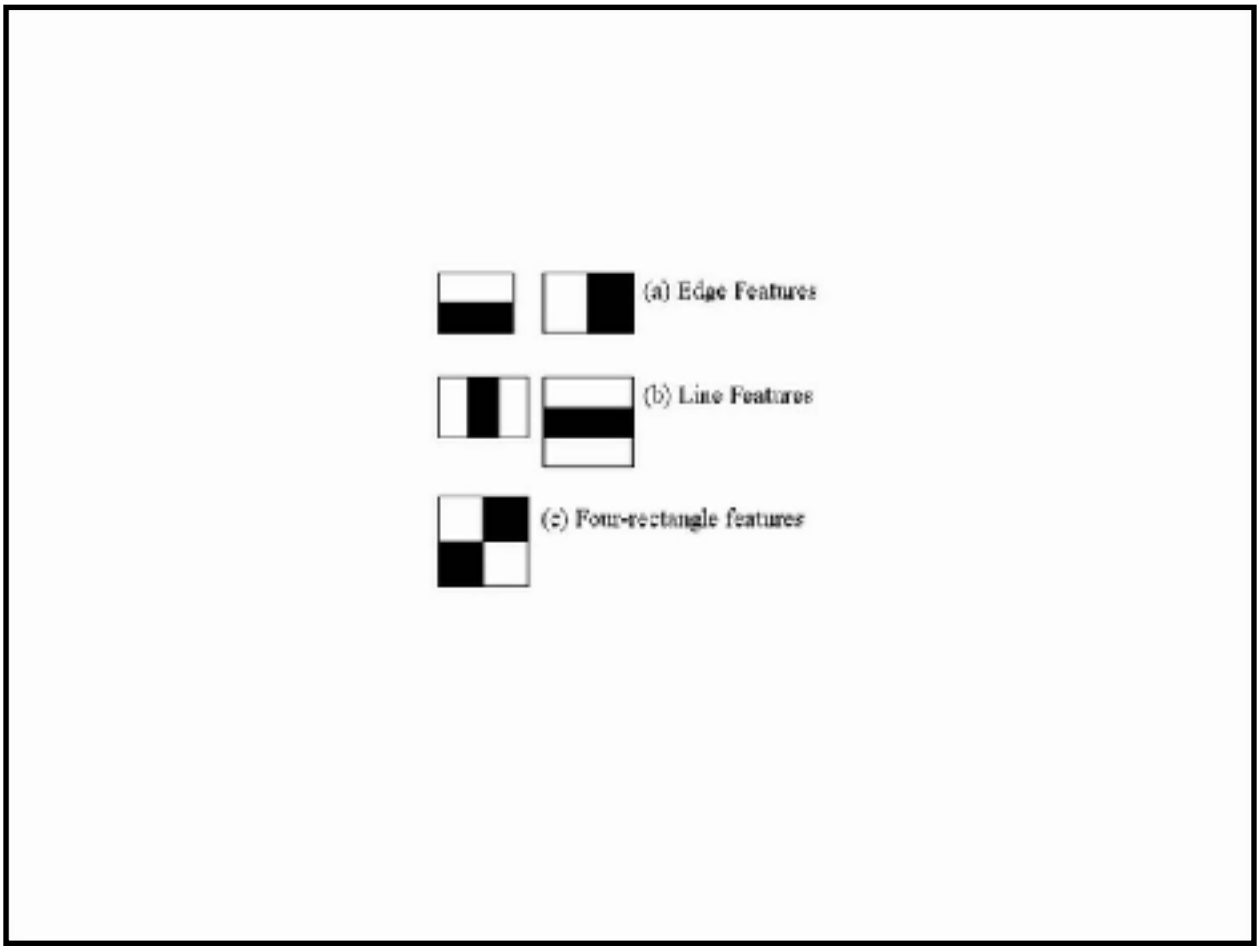
Face detection using Haar Feature-based Cascade Classifiers:

Version 1

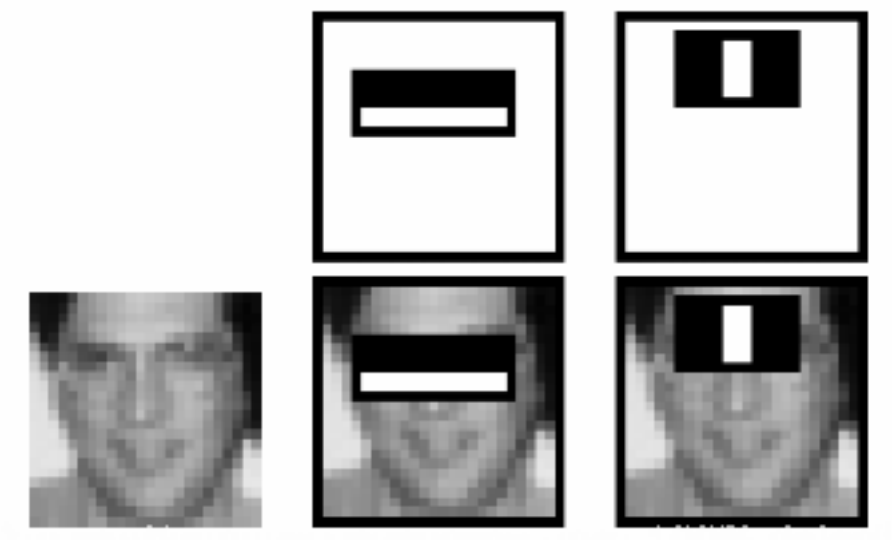
1. Object detection using Haar feature-based cascade classifiers is an effective object detection method proposed by ***Paul Viola*** and ***Michael Jones***.
2. A machine learning based approach where a cascade function is trained from a lot of **positive** and **negative** images. It is then used to *detect objects* in other images.
3. Here we will work with face detection - initially, algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it.
   - Positive images - images of face
   - Negative images - images without face
4. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under the black rectangle.
5. For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces calculations for a given pixel to an operation involving just four pixels.

Consider the image below 1.2. The top row shows two good features:

1. First feature focus on the property that region of eyes is often darker than the region of the nose and cheeks.
2. Second feature relies on the property that eyes are darker than the bridge of the nose.



Adaboost

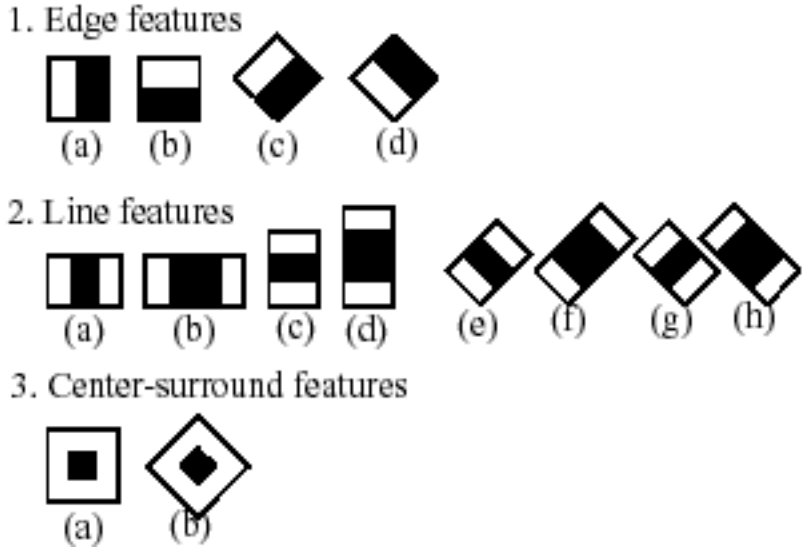


1.1

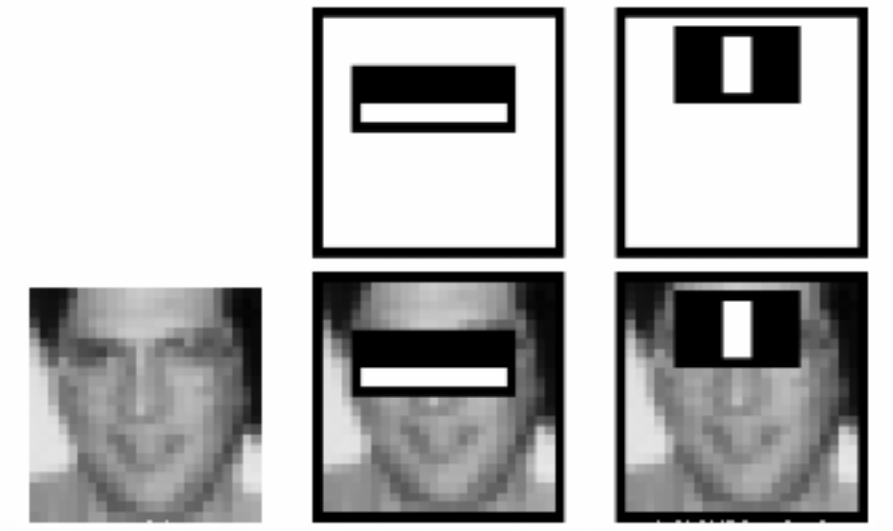
1.2

Version 2:

Haar classifier is a machine learning based approach , an algorithm created by Paul Viola and Michael Jones.
Trained from many positive images (with faces) and negative images (without faces).



It starts by extracting Haar features from each image by above windows. Each window is placed on the picture to calculate a single feature. This feature is single value obtained by subtracting sum of pixels under white part of the window from the sum of pixels under the black part of the window.
Now, all possible sizes of each window are placed on all possible locations of each image to calculate plenty of features.



For example, in above image, we are extracting two features.

1. First one focuses on the property that the region of the eyes is often darker than the area of nose and cheeks.
2. The second feature relies on the property that the eyes are darker than the bridge of nose.

Also most of the features are irrelevant. For example, when used on cheek, the windows become irrelevant because none of these areas are darker or lighter than other regions on the cheeks, all sectors here are the same. So we promptly discard irrelevant features and keep only those relevant with a fancy technique called **Adaboost**.

Adaboost is a training process for face detection, which selects only those features known to improve the classification (face / non-face) accuracy of our classifier.

In the end, most of the region of an image is a non-face region. Considering this, it's a better idea to have a simple method to check if a window is a non-face region, and if it's not, discard it right away and don't process it again. So we can focus mostly on the area where the face is.

OpenCV:

- OpenCV is the most popular library for computer vision.
- OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn't one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched.
- The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called Classifiers.
- OpenCV uses cascades. What's a cascade? The best answer can be found in the dictionary: " a waterfall or series of waterfalls. " Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages.
    - For each block, it does a very rough and quick test.
    - If that passes, it does a slightly more detailed test, and so on.
- The algorithm may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass.

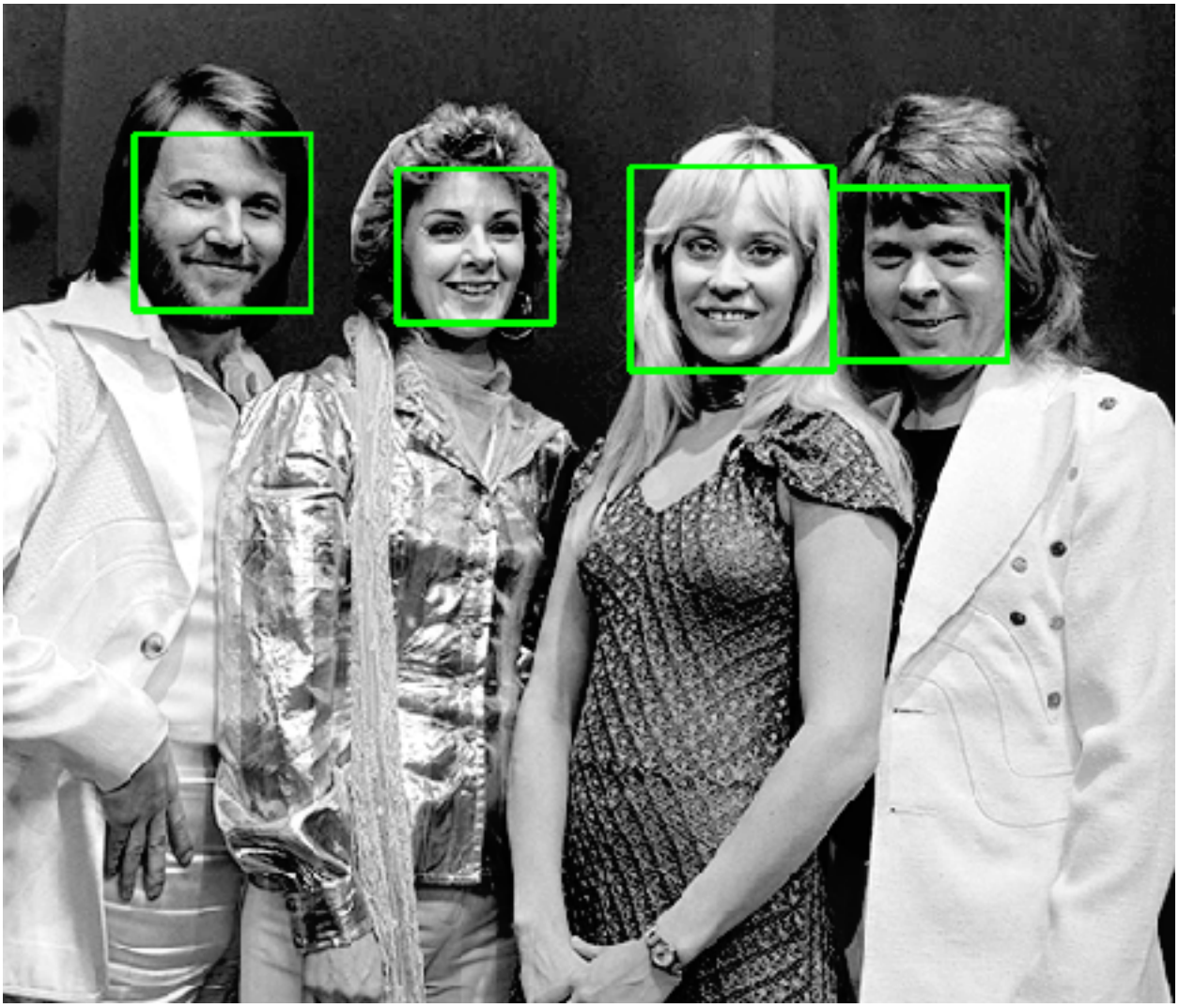
The advantage is that the majority of the picture will return negative during the first few stages, which means the algorithm won't waste time testing all 6000 features on it.

WITH STATIC IMAGE

```
# import OpenCV library
import cv2
Import sys

# Get user supplied values for image - png and cascade - xml (of faces)
imagePath = "abba.png"
cascPath = "haarcascade_frontalface_default.xml"

# Create the haar cascade
# Now we create the cascade and initialize it with out face cascade. This loads the face cascade into memory. cascade is just an XML file that contains data to detect faces.
faceCascade = cv2.CascadeClassifier(cascPath)

# Read the image - Here we read the image and convert it to grayscale, as many operations in OpenCV are done in grayscale.
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor = 1.1,
    minNeighbors = 5,
    minSize = (30, 30),
    flags = cv2.CASCADE_SCALE_IMAGE
)
# detectMultiScale is a general function that detects objects.
# -- first option is grayscale image.
# -- second is scaleFactor -> since some faces may be closer to camera, they would appear bigger than the faces in the back. The scale factor compensates for this.
# -- minNeighbors -> how many objects are detected near the current one before it declares the face found.
# -- minSize -> size of each window.

This function returns a list of rectangles in which it believes it found a face.

print "Found {0} faces!".format(len(faces))

# Draw a rectangle around the faces.

for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow("Faces found", image)
cv2.waitKey(0)

We display the image and wait for the user to press a key.
```

OUTPUT:

```
Object-Detection — -bash — 80×33
Last login: Sat Jul 28 18:21:33 on ttys001
Anaghas-MBP:Object-Detection anaghakelkar$ python opencv_test.py
libpng warning: iCCP: profile 'Photoshop ICC profile': 'GRAY': Gray color space
not permitted on RGB PNG
Found 4 faces!
Anaghas-MBP:Object-Detection anaghakelkar$
```



WITH LIVE VIDEO

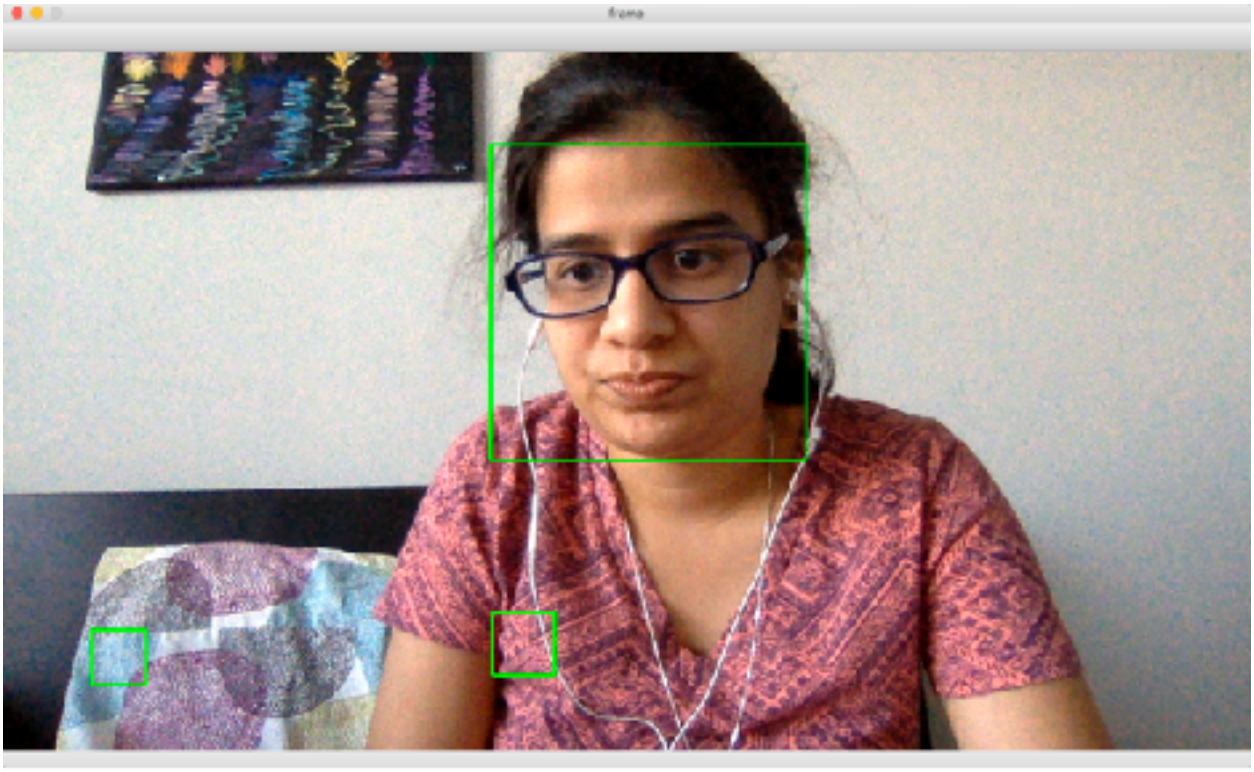
```
# capture live video
cap = cv2.VideoCapture(0)

while(True):
        ret, frame = cap.read() # read each frame of video
        # Run haar classifier on each frame and detect faces
        if cv2.waitKey(1) & 0xFF == ord('q'):
                break
# when everything is done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Output:

```
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 2 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
```

Found 1 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 2 faces!
Found 1 faces!
Found 1 faces!
Found 2 faces!
Found 0 faces!
Found 0 faces!
Found 1 faces!
Found 2 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 2 faces!
Found 1 faces!
Found 2 faces!
Found 3 faces!
Found 2 faces!
Found 1 faces!
Found 1 faces!
Found 4 faces!
Found 2 faces!
Found 1 faces!
Found 3 faces!
Found 1 faces!
Found 2 faces!
Found 3 faces!
Found 3 faces!
Found 3 faces!
Found 2 faces!
Found 2 faces!
Found 3 faces!
Found 2 faces!
Found 3 faces!
Found 2 faces!
Found 3 faces!
Found 4 faces!
Found 5 faces!
Found 4 faces!
Found 2 faces!
Found 2 faces!
Found 2 faces!

Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 2 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
Found 1 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
Found 1 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
Found 1 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 2 faces!
Found 1 faces!
Found 2 faces!
Found 2 faces!
Found 2 faces!
Found 2 faces!
Found 1 faces!


frame

Wink Detection Program:

```
import sys
import cv2
import numpy as np
import os
from os import listdir
from os.path import isfile, join

def detectWink(frame, location, ROI, cascade):
    eyes = cascade.detectMultiScale(ROI, 1.27 ,7 , 0|cv2.CASCADE_SCALE_IMAGE, (10, 10))
    if len(eyes) == 0:
        eyes = cascade.detectMultiScale(ROI, 1.02 , 3 , 0|cv2.CASCADE_SCALE_IMAGE, (10,
10))
    for e in eyes:
        e[0] += location[0]
        e[1] += location[1]
        x, y, w, h = e[0], e[1], e[2], e[3]

        cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 0, 255), 2)
    return len(eyes) == 1

def detect(frame, face_cascade, eye_cascade):
    scaleFactor = 1.25
    minNeighbors = 3
    flag = 0|cv2.CASCADE_SCALE_IMAGE
    minSize = (30,30 )
    faces = face_cascade.detectMultiScale(
        frame,
        scaleFactor,
        minNeighbors,
        flag,
        minSize)
    detected = 0
    for f in faces:
        x, y, w, h = f[0], f[1], f[2], f[3]
        faceROI = frame[y:y+h, x:x+w]
        if detectWink(frame, (x, y), faceROI, eye_cascade):
            detected += 1
            cv2.rectangle(frame, (x,y), (x+w,y+h), (255, 0, 0), 2)
        else:
            cv2.rectangle(frame, (x,y), (x+w,y+h), (0, 255, 0), 2)
    return detected

def runonFolder(face_cascade, eye_cascade, folder):
    if folder[-1] != "/":
        folder = folder + "/"
    files = [join(folder, f) for f in listdir(folder) if isfile(join(folder, f))]
```

```
    windowName = None
    totalCount = 0
    for f in files:
        img = cv2.imread(f, 1)
        if type(img) is np.ndarray:
            icnt = detect(img, face_cascade, eye_cascade)
            totalCount += icnt
            if windowName != None:
                cv2.destroyWindow(windowName)
            windowName = f
            cv2.namedWindow(windowName, cv2.WINDOW_AUTOSIZE)
            cv2.imshow(windowName, img)
            cv2.waitKey(2000)
    return totalCount

if __name__=='__main__':
    if len(sys.argv)!=1 and len(sys.argv)!=2:
        print(sys.argv[0] + " :got " + len(sys.argv)-1 + " arguments. Expecting 0 or 1 : [image-
folder]")
        exit()

    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+
'haarcascade_frontalface_default.xml')
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+ 'haarcascade_eye.xml')

    if len(sys.argv)==2:
        folderName = sys.argv[1]
        detections = runonFolder(face_cascade, eye_cascade, folderName)
        print("Total of ", detections, " detections")
    else:
        runonVideo(face_cascade, eye_cascade)
```

Output:

Anaghas-MBP:Object-Detection anaghakelkar$ python DetectWink.py WinkImages/

('Total of ', 21, ' detections')

 # run on video

```
def runonVideo(face_cascade, eye_cascade):
    videocapture = cv2.VideoCapture(0)
    if not videocapture.isOpened():
        print("Can't open video default camera!")
        exit()
```

```
windowName = "Live Video"
showlive = True
while(showlive):
    ret, frame = videocapture.read()
    if not ret:
        print("Can't capture frame")
        exit()
    cnt = detect(frame, face_cascade, eye_cascade)
    print("Detections: {0}".format(cnt))
    cv2.imshow(windowName, frame)
    if cv2.waitKey(30)>=0:
        showlive = False
videocapture.release()
cv2.destroyAllWindows()
```